大規模ランダムスパース行列演算プログラムライブラリー開発の試み (数値計算のアルゴリズムの研究)

AUTHOR(S):

池田, 宏明; 大月, 邦俊

数理解析研究所講究録
310巻 1977年 1-18

大規模ランダムスパース行列演算プログラムライブラリ開発の試み

千葉大学工学部　池田宏明
大月邦俊

1. まえがき

行列の全要素数に対する非零要素数の割合が少なく，いわゆるスパース行列の処理においては，非零要素のみを演算の対象にすることにより，従来の行列処理に対してより少ない空間を用い，より少ない時間で演算できることが知られている。[1,2] 特に対称行列あるいは構造的に対称な行列に関しては，行列を表わすデータ構造を簡単にできることもあって，二，三の方法が提案されている。[3,4] 一方，帯状行列に対する行列演算ライブラリも開発され[5]，また帯状行列の仮想記憶方式計算機を意識した算法も発表された。[6]

これに対して筆者らはランダムなスパース行列の演算を行なうプログラムライブラリの開発を意図している。ランダムスパース行列の演算に関しては，ピボット選択法に関して検討されているが[7,8] いまだ国内では実用に供しうるようなプ

ログラムは開発されていないと考えられる。また，ランダム行列に対してスパース行列処理をしたために生ずる時間と空間の減少量についても具体的には与えられていない。この試みは大規模なランダムスパース行列を，直接ガウス消去法あるいはLU分解法によって処理しようとするものであり．初期行列の非零要素値およびその属性（行番号，列番号）のみを入力とし，行列の内部表現のデータ構造として行方向および列方向のリンク付リスト構造を用いる。このデータ構造を用いて，ランダムスパース行列を係数行列とする連立方程式の求解および行列どうしの加減算，乗算などをプログラミングした場合の所要メモリおよび演算の手数について検討し，特に演算手数については，行列の次数に対して一次すなわちオーダーの手数で十分であることを示す。

現在，加減算，乗算およびGauss-Jordan法ならびにLU分解法を用いた連立方程式の求解のプログラムは一応完成しているが，本報告は，昭和51年度に開始したプログラムライブラリ開発の途中経過について記述したもので不十分な点があることを恐れる。諸氏の御批判をいただきたい。

2. 行列を表現するデータ構造

2.1 リンク付リスト構造と所要空間　大量のデータを効率よく処理するためには，それぞれの問題に適したデータ

構造を用いることが必要である。対称行列あるいは構造的に対称な行列を記述するデータ構造としては比較的簡単なものが用いられている。[3,4] しかし，一般のランダム行列に対しては，(1)非零要素の値を行（あるいは列）ごとにまとめ，これを一次元配列に表現し，それぞれの要素の列番号（あるいは行番号）を別の一次元配列で与える方法，および(2)非零要素値とその位置を表わす行番号・列番号に対して，行方向および列方向にポインタを用いたリンク付リスト構造が用いられる。[1]※ 前者は後者に比較して所要空間が少なくてすむが，非零要素の追加あるいは削除に対応して，そのデータ構造の更新に多大の手数を要するという欠点を有する。そこで筆者らは後者のデータ構造を採用した。

例を用いて，リンク付リスト構造を図1に示す。"R"で始まる配列名は行方向のリストに関するものであり，"C"で始まる配列名は列方向のリストに関するものである。通常，この種のデータ構造では，要素の追加ならびに削除を容易に行ないうるように，逆方向ポインタおよび末尾ポインタを用いるが，ここではこれらを使用せず，所要空間の節約をはかった。これは，新しい要素はリンク付リストの先頭に容易に追

※ このデータ構造は，行番号，列番号と非零要素番号をそれぞれ有向グラフの始点，終点と枝番号に対応づければ，有向グラフを表現するデータ構造と一致する。

$$A_0 = \begin{bmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 2 & 2.5 & 4 & 2.6 \\ 2 & 0 & 3 & 0 & 0 \\ 1.5 & 0 & 2.5 & 4 & 0 \\ 3 & 1 & 0 & 0 & 5 \end{bmatrix}$$

V: 非零要素値
R: 行番号, RP: 行方向ポインタ, RH: 行ヘッド
RN: 行の非零要素数
C: 列番号, CP: 列方向ポインタ, CH: 列ヘッド
CN: 列の非零要素数

(行番号)	1	2	3	4	5
RH	1	4	8	10	13

(行番号)	1	2	3	4	5
RN	3	4	2	3	3

(非零要素番号)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RP	2	3	0	5	6	7	0	9	0	11	12	0	14	15	0
R	1	1	1	2	2	2	2	3	3	4	4	4	5	5	5
V	1.0	3.0	2.0	2.0	2.5	4.0	2.6	2.0	3.0	1.5	2.5	4.0	3.0	1.0	5.0
C	1	4	5	2	3	4	5	1	3	1	3	4	1	2	5
CP	8	6	7	14	9	12	15	10	11	13	0	0	0	0	0

CH	1	4	5	2	3
(列番号)	1	2	3	4	5

CN	4	2	3	3	3
(列番号)	1	2	3	4	5

図1. 行列とそれを表わすデータ構造 ($N=5$, $NZ_0=15$)

加でき，また後に示すように，要素の削除を要する行あるいは列については，対応するリストを走査する必要があり，これと同時に削除できることによる。一方，後に述べるピボット要素選択順序の確率論的決定に用いるために，各行・各列に存在する非零要素数を示すところの配列 RN と CN を用いている。行列の次数を N，非零要素数を NZ_0 とすれば，RH, RN, CH, CN はそれぞれ長さ N の配列であり，V, RP, R, CP, C はそれぞれ長さ NZ_0 の配列であるから，所要空間は $(4N+5NZ_0)$ である。

スパース行列の統計から，$NZ_0 = kN$, $k=2\sim10$ であるとされている。[9] これを用いて，N×N の正方行列を2次元配列を用いて表現した場合と図1のリスト構造を用いて表現した場合の空間量の比較を図2に示した。kは行（列）の平均非零要素数である。

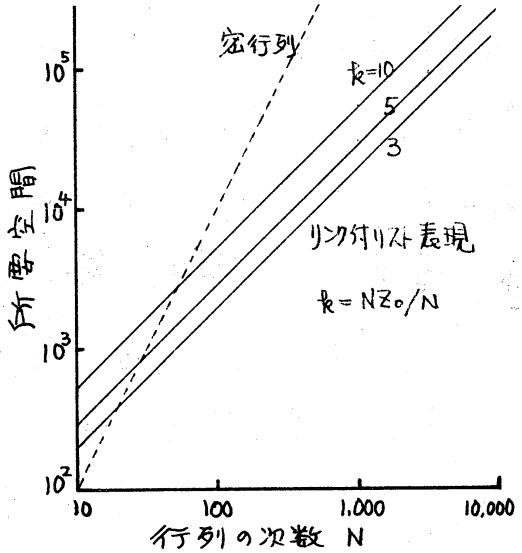

図2. 所要空間の比較

2.2 データ構造の作成と所要手数

図1のデータ構造は，入力データ V,R,C から次に示す方法で作成できる。ステップ 3,4,5 は行方向リンク付リストの作成であり，ステップ 6,7,8 は列方向リンク付リストの作成である。ここで，配列

```
begin
1.    RH=0; RN=0; RP=0; CH=0; CN=0; CP=0.
2.    非零要素番号 I のすべてについて以下を行なう.
      begin
3.      RN(R(I))=0 であれば, RH(R(I))=I として ステップ 5 へ.
4.      さもなくば, RP(RN(R(I)))=I
5.      RN(R(I))=I
6.      CN(C(I))=0 であれば, CH(C(I))=I として ステップ 8 へ.
7.      さもなくば, CP(CN(C(I)))=I
8.      CN(C(I))=I
      end
9.    RN=0; CN=0.
10.   非零要素番号 I のすべてについて以下を行なう.
      begin
11.     RN(R(I))=RN(R(I))+1
12.     CN(C(I))=CN(C(I))+1
      end
end
```

RN, CN は作業用に用いられており，ステップ 9～12 で各行・各列の非零要素数が代入される。必要な手数は O(N+NZ。) であることは明らかであろう。

3. 連立一次方程式の解を求めるプログラム

リンク付リスト構造を用いて次数Nのランダム実行列Aを係数行列とする連立方程式 $Ax=b$ の解 x を求めるプログラムを FORTRAN を用いて作成した。なお，求解の算法として，Gauss-Jordan 法と LU 分解法 (Doolittle 法[4]) を採用した。

3.1 作業用配列の有効利用　リンク付リスト構造はそれ自身効率よく行方向に，あるいは列方向に非零要素を探索できるものであるが，行列演算において，ある行（列）と異なる列あるいは行の間で演算を繰返えす場合には，その手数が従来の密行列演算の手数と同程度となり，スパース行列処理の利点がなくなる。そこで筆者らは別に長さNの作業用配列 WORK を用意し，いま着目している行（列）についてまずポインタをたどり対応する行（列）の非零要素配置をこれに再現しておく。この後被演算列（行）の行番号あるいは列番号に応じて WORK を参照すれば，直ちに所定の演算が可能になる。

3.2 Gauss-Jordan 法を用いた解法　プログラムの流れを示せば次のようになる。ここで B は $Ax=b$ の b に相当する長さNの定数ベクトルであり，解は B に求められる。

[Gauss-Jordan 法]

```
  begin
1.  各ピボット要素番号 JP について以下を行なう
        begin
2.      JP を除く C(JP)列の各要素 J について以下を行なう.
            begin
3.              B(R(J))=B(R(J))-B(R(JP))・V(J)/V(JP)
4.              R(J) 行の各要素 K について以下を行なう.
                    begin
5.                      WORK(C(K))=K とし R(J) 行の非零要素配置を再現すると
                        同時に R(J) 行のリング付リストから J を削除する
                    end
6.              R(J) 行の要素数を 1 減ずる
7.              削除した要素番号 J を free-list の先頭に追加する
8.              R(JP) 行の要素 L について以下を行なう.
                    begin
9.                      もし WORK(C(L))≠0 であれば,
                          V(WORK(C(L)))=V(WORK(C(L)))-V(L)・V(J)/V(JP);
10.                     さもなくば,-V(L)・V(J)/V(JP) を fill-in として リストに加える.
                    end
11.             R(J) 行の fill-in を除く各要素 K について以下を行なう.
                    begin
12.                     WORK(C(K))=0
                    end
            end
13.     JP のみを残し, C(JP)列の全ての要素を削除する.
        end
14. 1≦i≦N なる i 行について以下を行なう.
        begin
15.     B(i)=B(i)/V(RH(i))
        end
  end
```

Step 1. 各段階でのピボット要素番号JPは4. でのべるところの subroutine によって fill-in ができるだけ少なくなるものに決定される。

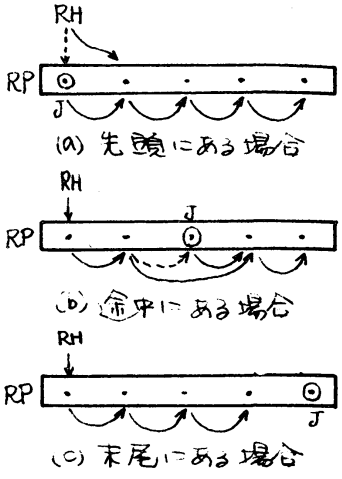

図3. 要素Jの削除

Step 5. ここで採用しているデータ構造では，逆方向ポインタを持っていないので，リストの途中あるいは末尾にある要素を削除するには，その要素に到達するまで行方向にリストをたどらなければならない。しかしこの行に関する非零要素配置を WORK に再現するためにリストをたどるから，これと同時に所定の要素を削除できる。この様子を図3に示した。先頭にある場合は RH を変更するだけでよい。途中あるいは末尾にある場合は "RP(K)=J ならば RP(K)=RP(J)" を実行することによりJを削除できる。点線は変更された箇所を示している。この部分の実行回数は $O(N \cdot NZ)$ である。

Step 7. free-list の拡張。初め free-list の先頭ポインタ FREE は overflow 領域の先頭に等しくしておく。いま Step 5 で J が消去されると，"RP(J)=FREE; FREE=J" によって，これを free-list の先頭に加えることができる。free-list のリンクポインタとして RP を使用している。図4にJが追加される様子を示した。overflow 領域の先頭を常に0にしておくことによって，

free-list を使いはたしたかどうかを判定できる。Step 3, 6, 7 の実行回数はそれぞれ O(NZ) に等しい。

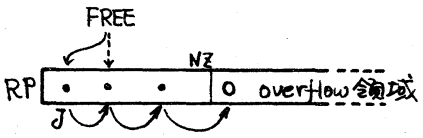

図4. free-list の拡張

Step 9, 10. R(J)行に関する WORK と R(JP) 行の要素の列番号とを比較して掃出しに伴なう要素値の修正を行なう。WORK(CL) =0 のとき R(J) 行, C(L)列に fill-in が発生する。free-list があればこれを使用し, free-list を使いはたしたとき overflow 領域を用いる。このことにより実質的な fill-in の数を減少している。このステップの実行回数は O(N·NZ) である。

Step. 12. 実行回数は Step 5 と同様に O(N·NZ) である。

Step 13, 15 の実行回数はそれぞれ O(N) であるから, リング付リスト構造により表現された行列に Gauss-Jordan 法を適用した場合の手数は O(N·NZ) であるといえる。なお. NZ は各消去段階での非零要素数であり, 初期行列では NZ=NZ$_0$ であるが, fill-in のため しだいに増加し, 最終的には NZ=N となる。

3.3. LU 分解法を用いた解法　　LU 分解の標準的な方法は内積累和法であるが, リング付リスト構造で表現された行列の LU 分解に適し, また fill-in が生ずるかどうかの判断が容易にできることから, ここでは次に示す LU 分解法[3,4]を採用した。すなわち, 行列 A の要素を a_{ij} とするとき,

1. 第1列をそのまま採用する。

2. 第1行のすべての非対角要素を対角要素で割る。
3. $i \geq 2, j \geq 2$ の要素について，$a_{ij} \cdot a_{ij}$ をそれから引く。
4. ステップ3で演算される部分行列の次数が2以上であれば，この部分行列に関してステップ1へもどる。さもなくば，終り。

リンク付リストで表現された行列にこの方法を適用した場合のプログラムの流れを次ページ(11ページ)に示した。ここで第k段階のLU分解の対象となる部分行列とは，a_{ij}, $i \geq k$ $j \geq k$ から成る行列のことである。

Step 1. 各段階でのピボット要素JPは，4.でのべる方法によって決定され，それが対角要素になるように並べかえてある。

Step 2. DIAGは長さNの補助配列で，対角要素であるピボット要素番号をこれに登録しておくことにより，求解における前進代入での対角要素の検索を省略することができる。O(N).

Step 5, 14 fill-inの個数は部分行列の各行，各列の非零要素数によって確率論的に決定される。Gauss-Jordan法のように掃出しを行なわないが，部分行列の要素数は減少する。このステップの実行回数はそれぞれ O(NZ) である。

Step 7. 部分行列の $i \geq 2, j \geq 2$ の要素 a_{ij} について $a_{ij} = a_{ij} - a_{ij} \cdot a_{ij}$ を効率よく実行するために，C(J)列について WORKに非零要素配置を再現している。必要な手数は O(N·NZ) である。

[LU分解法]

```
    begin
1.    各ピボット要素 JP について以下を行なう.
        begin
2.        DIAG(R(JP))=JP
3.        部分行列の第1行の JP を除く要素 J について以下を行なう.
            begin
4.            V(J)=V(J)/V(JP)
5.            C(J) 列の要素数を 1 減ずる.
6.            部分行列の C(J) 列の要素 M について以下を行なう.
                begin
7.                WORK(R(M))=M
                end
8.            部分行列の第1列の JP を除く要素 I について以下を行なう.
                begin
9.                WORK(R(I))≠0 であればV(WORK(R(I)))=V(WORK(R(I)))-V(J)・V(I)
10.               さもなくば -V(J)・V(I)を fill-in としてリストに加える。
                end
11.           部分行列の C(J) 列の fill-in を除く要素 M について以下を行なう.
                begin
12.               WORK(R(M))=0
                end
            end
13.       部分行列の第1列の JP を除く要素 I について以下を行なう.
            begin
14.           R(I) 行の要素数を 1 減ずる.
            end
        end
    end
```

Step 8,9　ステップ7で作成したWORKを参照しつつ，R(I)行 C(J) 列の要素の修正を行なう。

（a）　R(JP)行，C(JP)列にそれぞれJPを含めて2個以上の要素が存在する場合：WORK(R(I))=0であればfill-inが発生することがわかるから，overflow領域にR(I)行 C(J)列の要素を加え，それぞれの行，列の要素数を1増加する。fill-inの取扱いはGauss-Jordan法の場合と同様である。

（b）　R(JP)行の要素がJPのみである場合：直ちにステップ13以下を実行する。

（c）　C(JP)列の要素がJPのみである場合：次の段階へ進む。

この部分の実行回数は O(N·NZ) である。

Step 12.　WORKのクリア。手数は O(N·NZ)。

このようにしてLU分解が完了する。したがって，リング付リスト構造により表現された行列をLU分解するに要する手数は O(N·NZ) であるといえる。ここで NZ は各段階での部分行列の非零要素数であり，初期行列では NZ=NZ。であるが，fill-inのためしだいに増加し，最終的には NZ=1 となる。

解を求めるためにはさらに，前進代入および後退代入操作を行なう。この操作は次に示すようにプログラミングでき，必要な手数は O(NZ) である。

```
    begin
1.    1≦k≦N-1 について以下を行なう.
        begin
2.        B(k)=B(k)/V(DIAG(k))
3.        k列の要素Iのうち R(I)＞k を満すIについて以下を行なう.
            begin
4.            B(R(I))=B(R(I))-B(k)・V(I)
            end
        end
5.    B(N)=B(N)/V(DIAG(N))
6.    1≦k≦N-1 について以下を行なう.
        begin
7.        (N+1-k)列の要素Iのうち R(I)＜(N+1-k)を満すIについて以下を行なう.
            begin
8.            B(R(I))=B(R(I))-B(N+1-k)・V(I)
            end
        end
    end
```

4. ピボット要素の選択手法

4.1 いろいろな手法とその手数　スパース行列の演算の過程で fill-in が多数発生すると NZ が増大する。この結果初期行列のスパース性が失なわれ，スパース処理の利点が損なわれることになる。fill-in の発生量はピボットの選択順序に依存しており，ピボット選択はこの意味から重要である。その手法としては[7,8,10]

(1) 決定論的手法：各段階での行列の非零要素に対して，実際の fill-in の個数を計算し，これが最小となる要素をピボットに選ぶ手法である。この手法はグラフ理論との関連において検討されているが，表1に示したように多くの手数を要する。

表1. ピボット選択手法とその手数

手法	決定論的手法	確率論的手法			
	$A^{(k)}$ について実際の fill-in を計算する	RN最小のものから選ぶ		RN·CN 最小のものから選ぶ	
		$A^{(0)}$ のみ	$A^{(k)}$ について	$A^{(0)}$ のみ	$A^{(k)}$ について
手数	$O(N\cdot NZ^2)$	$O(NZ_0)$	$O(N\cdot NZ)$	$O(NZ_0)$	$O(N\cdot NZ)$

(2) 確率論的手法：ピボットの候補のそれぞれに対して fill-in を発生する可能性の最大値を計算し，それが最小となる要素をピボットに選ぶ手法である。この手法には，(a) 初期行列 $A^{(0)}$ の各行の非零要素数について，$RN(i_1) \le RN(i_2) \le \cdots \le RN(i_N)$ が成立するとき，$i_1, i_2, \cdots, i_N$ 行から順にピボットを選ぶ方法，(b) $A^{(0)}$ について，(a)の方法に加えてさらに列方向の非零要素数 $CN(j)$ が最小となる要素をピボットにする方法，さらに (c) 上の (a), (b) を各段階での行列 $A^{(k)}$ に適用する方法がある。ここで，$RN(i)$, $CN(j)$ はそれぞれ，i 行と j 列の要素数である。筆者らはピボット選択に要する手数が，3. でのべた掃出し操作あるいは LU 分解操作に要する手数を上まわらず，しかも fill-in が最小になるという観点から $A^{(k)}$ について RN, CN 両方を考慮した方法を採用した。

4.2 Gauss-Jordan 法の場合　ピボット要素 JP 選択の基本的な考え方は文献(8)と同一であるが，Gauss-Jordan 法では free-list を fill-in のためにあてることができるので，これを考慮して本法に適した確率論的手法を用いた。すなわち，ピボット要素番号が JP であるとき，fill-in の個数の最大値 F_{max} は，

$$F_{max} = (RN(i)-1)(CN(j)-1) - (CN(j)-1)$$
$$= (RN(i)-2)(CN(j)-1).$$

ここで $i=R(JP)$, $j=C(JP)$ であり，各段階で $F^{(k)}_{max}$ を最小にする要素 $JP^{(k)}$ をピボットに選ぶ。

4.3 LU分解法の場合　　文献(8)の方法をそのまま用いた。

5. ランダム行列を用いた実行結果

リンク付リスト構造を用いて連立方程式を解く二つの方法を示し，その手数は共に $O(N\cdot NZ)$ であることを得たが，これを確認するために両方法を FORTRAN を用いてプログラミングし，ランダムスパース行列を係数とする連立方程式を解いてみた。一行当りの非零要素数 k をパラメータとして実行結果を図5 (a), (b) に示した。これらの二方法による求解時間はほぼ等しく，共に $O(N^2)$ である。$NZ=kN$ とすれば，$O(N\cdot NZ)=O(N^2)$ であるから，演算中に fill-in が発生しているにもかかわらず，全体として見積った手数が保たれていると結論できる。所要空間も両方法ともほぼ等しい。図6 (a), (b) にはピボット要素選択に要する時間ならびに fill-in の個数を示したが，ピボット要素選択の手数も $O(N\cdot NZ)$ であることがわかる。

6. あとがき

fill-in を最小にするという基準で選んだピボットは必ずしも解の精度を保証しない。したがって，この種のプログラム

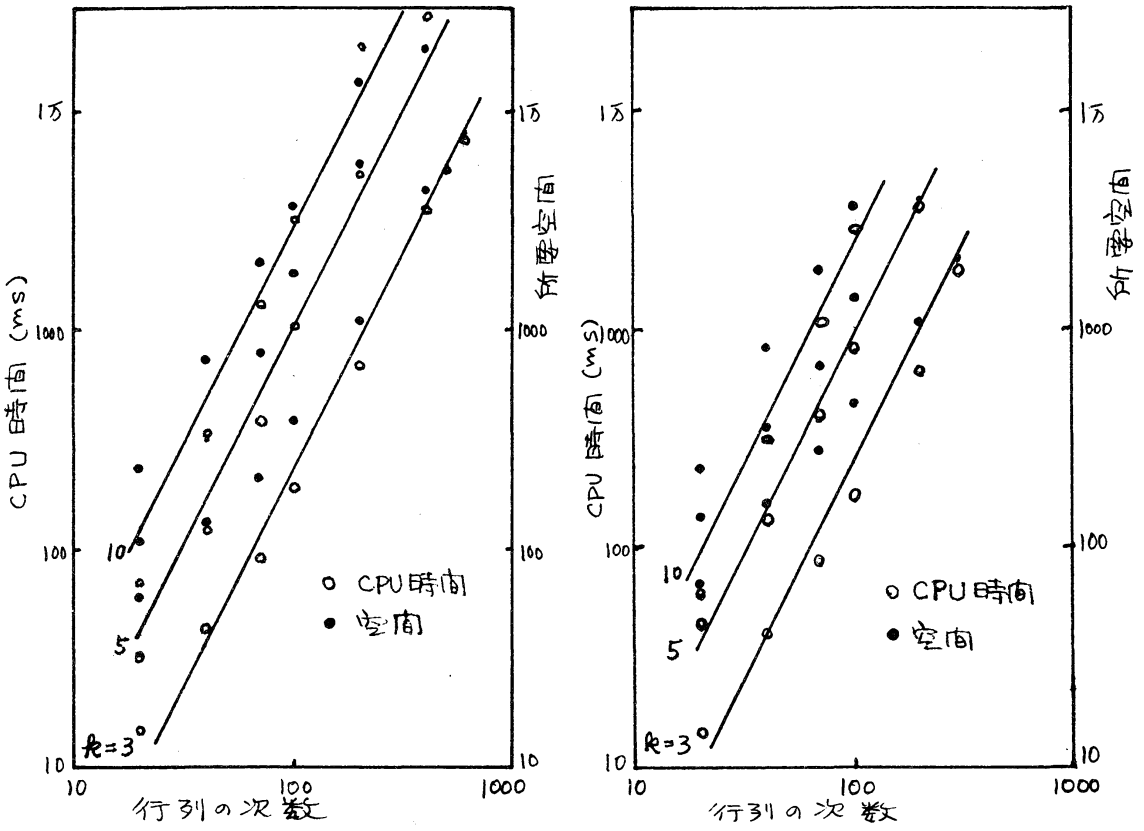

図5. 求解に要する時間と空間. ($k = NZ/N$)

を実用化するためには, (1)求められた解を初期値として反復改良法を適用する, (2) ピボットの値が小さいとき, これに適当な数を加えてから計算を実行し, 求められた解を修正するいわゆるピボット修正法[2]を用いる, などの方法を講じなければならない。

スパース行列どうしの加減算および乗算を実行するプログラムも作成した。この結果必要な手数はそれぞれ $O(N+NZ)$ および $O(N \cdot NZ)$ であり, 連立方程式の求解も含めて, 従来の密行列演算の手法に比較して N に対するオーダーを1次減ずる

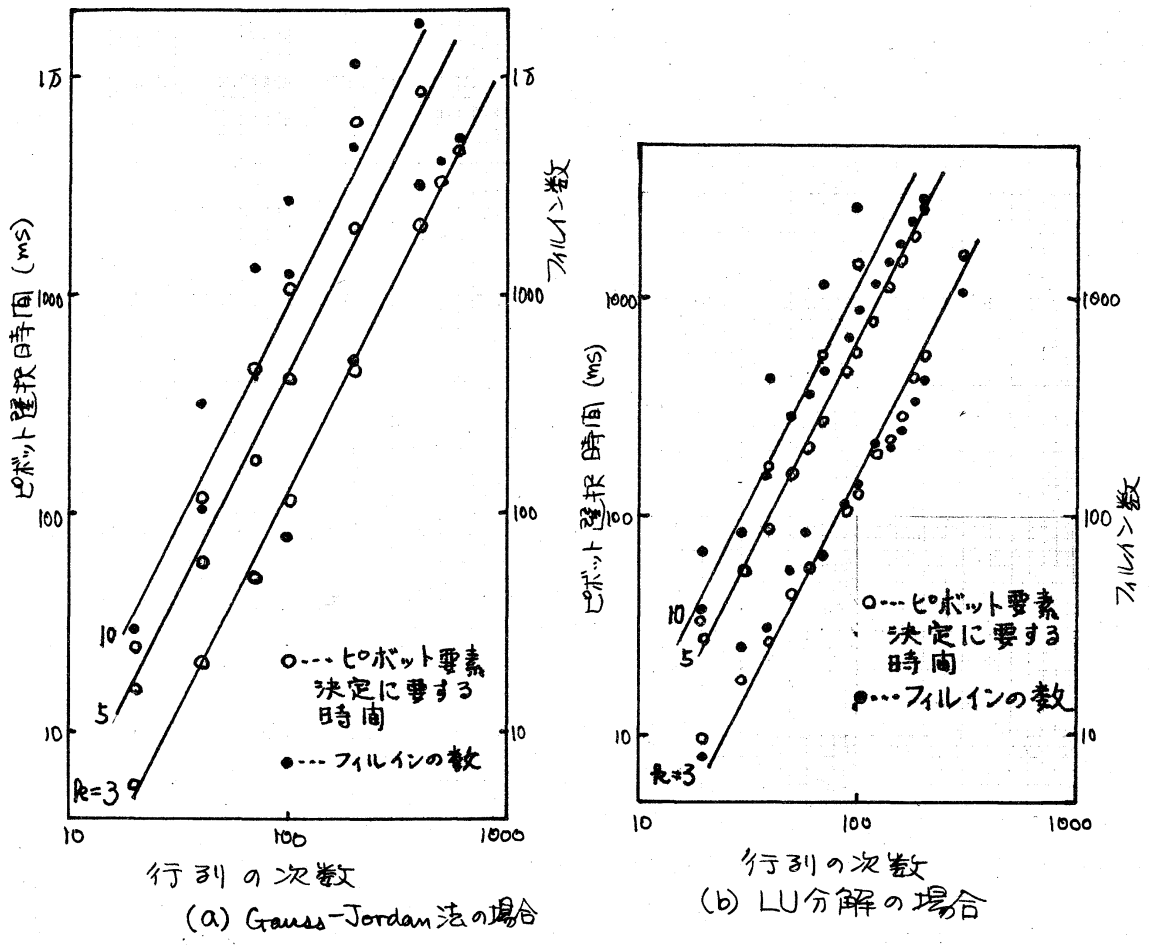

図6. ピボット要素決定に要する時間と fill-in の数

ことが可能なことを明らかにした。

本文では算法として直接法を用いたが，Gauss-Seidel 法・共役傾斜法などの間接法においては，解に収束するために係数行列Aに条件があるものの，非零要素数は不変であるという特長をもっている。また，これらの間接法の基本的演算は行列とベクトルの乗算であり，行列を表現するデータ構造も非零要素値とその属性（VとR,C）のみで十分である。上記2種の間接法についてもプログラムを作成し，100元～3000元のラ

ンダム行列とその対角要素を1に正規化した行列について解を求めた。この結果，求解に要するCPU時間は次式で与えられることを得た。（東大センター HITAC 8800/8700 システム使用）

$$T(ms) \fallingdotseq (\text{非零要素数}) \times (\text{繰返し回数}) / 100.$$

〔謝辞〕発表の機会を与えていただいた電気通信大学 名取亮先生に感謝する。また研究集会で有益な討論をしていただいた方々，日ごろ御援助をいただいている本学倉田是助教授に感謝する。東京大学大型計算センターから開発補助金を受けた。

文　献

(1) 伊理，中森：算法の最近の進歩，信学会誌 58，4，433-445，(1975-4)

(2) 大附，川北：スパース行列処理技法 (1)，(2)，(3)，情報処理，17，1，2，3

(3) R. D. Berry: An Optimal Ordering of Electronic Circuit Equations for a Sparse Matrix Solution, IEEE Trans. Circuit Theory, 40-50 (1971-1)

(4) L. O. Chau, P.M. Lin: Computer Aided Analysis of Electronic Circuits, Prentice Hall (1975)

(5) 日立製作所：OS7 数値計算ライブラリV，圧縮型線形計算 8700-7-005 (1976-2)

(6) 村田，堀越：対称帯行列を三重対角化するための新アルゴリズム 情報処理，16，2，93-101，(1973-9)

(7) 椎野：スパース行列の確率的考察，情報処理，14，9，677-683，(1973-9)

(8) H.Y. Hsieh: Pivoting-Order Computation Method for Large Random Sparse Systems, IEEE CAS-21,2, (1974-3)

(9) R.P. Tewarson: Sparse Matrices, Academic Press (1973)

(10) 坂元，白川，尾崎：スパースな連立方程式におけるピボット操作の順序づけ，情報処理，13，3，154-160，(1972-3)

(11) 池田，大月：非零要素のみを対象とするスパース行列演算の一方式，昭和51年 情報処理学会 大会 No. 205